GRID Superscalar: a programming model for the Grid

Daniele Lezzi, Oriol Jorba

Barcelona Supercomputing Center

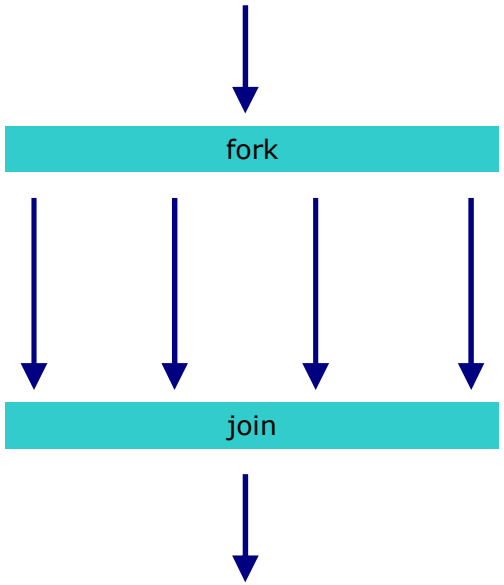
GOESSP Community Workshop, Hamburg, October 7

2009

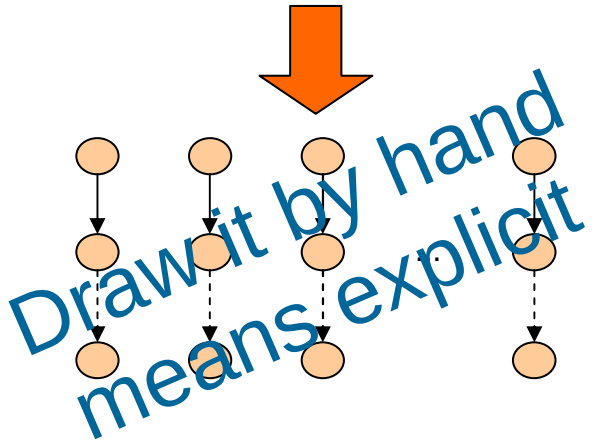
Motivation

- Can I run my programs in parallel?

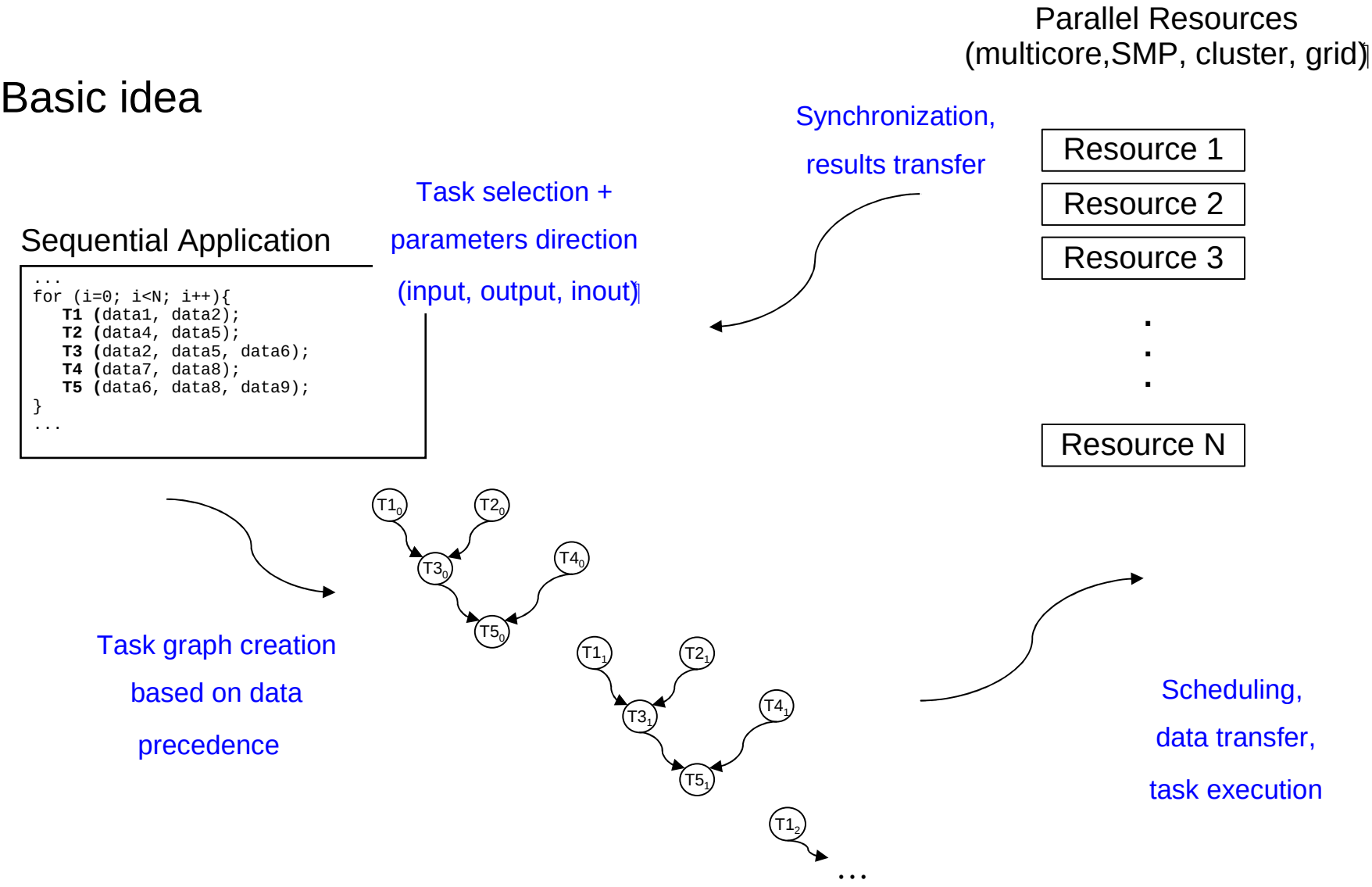
Explicit parallelism vs. Implicit parallelism




```
for(i=0; i < MSIZE; i++)
  for(j=0; j < MSIZE; j++)
    for(k=0; k < MSIZE; k++)
      matmul(A(i,k), B(k,j), C(i,j))
```

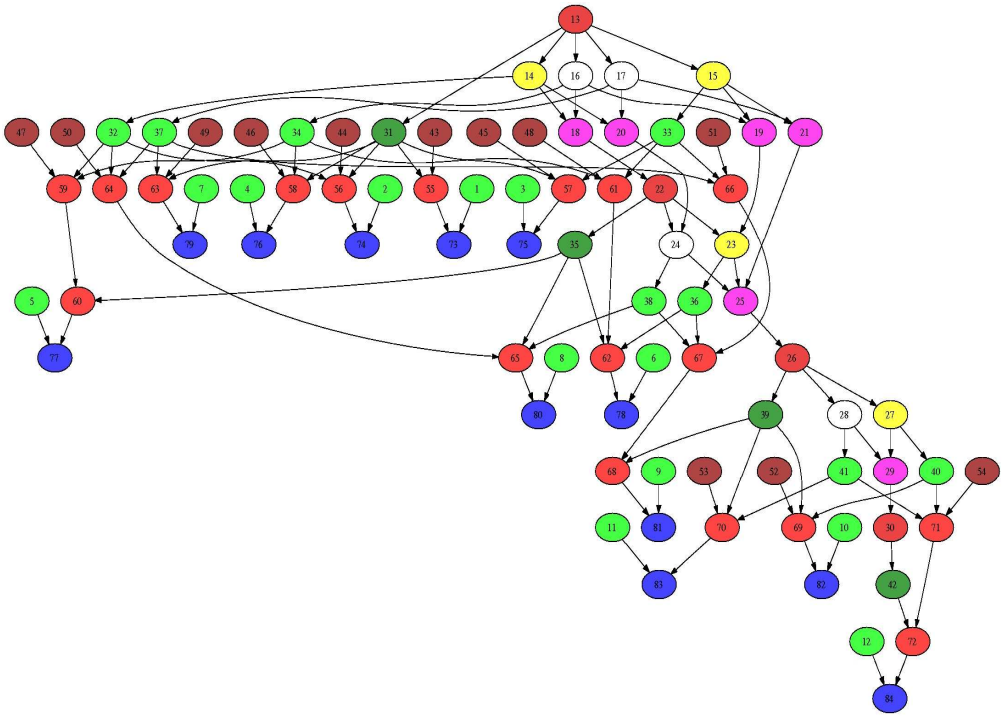


- Basic idea


Parallel Resources
(multicore,SMP, cluster, grid)
Synchronization,
results transfer
Resource 1
Resource 2
Resource 3
Resource N
Task selection +
parameters direction
(input, output, inout)
Sequential Application
...
for (i=0; i<N; i++){
T1 (data1, data2);
T2 (data4, data5);
T3 (data2, data5, data6);
T4 (data7, data8);
T5 (data6, data8, data9);
}
...
T1$_0$
T2$_0$
T3$_0$
T4$_0$
T5$_0$
T1$_1$
T2$_1$
T3$_1$
T4$_1$
T5$_1$
T1$_2$
Task graph creation
based on data
precedence
Scheduling,
data transfer,
task execution

Motivation

- Main objective: Reduce the complexity of applications development
  - Complexity of writing an application for a parallel platform comparable to writing it for a sequential platform
- Main characteristics
  - Task: unit of parallel work
  - Non intrusive programming model
  - Data dependence detection
  - Data renaming
  - Exploitation of distant parallelism



Design

```
for(i=0; i < MSIZE; i++)
  for(j=0; j < MSIZE; j++)
    for(k=0; k < MSIZE; k++)
      matmul(A(i,k), B(k,j), C(i,j))
```

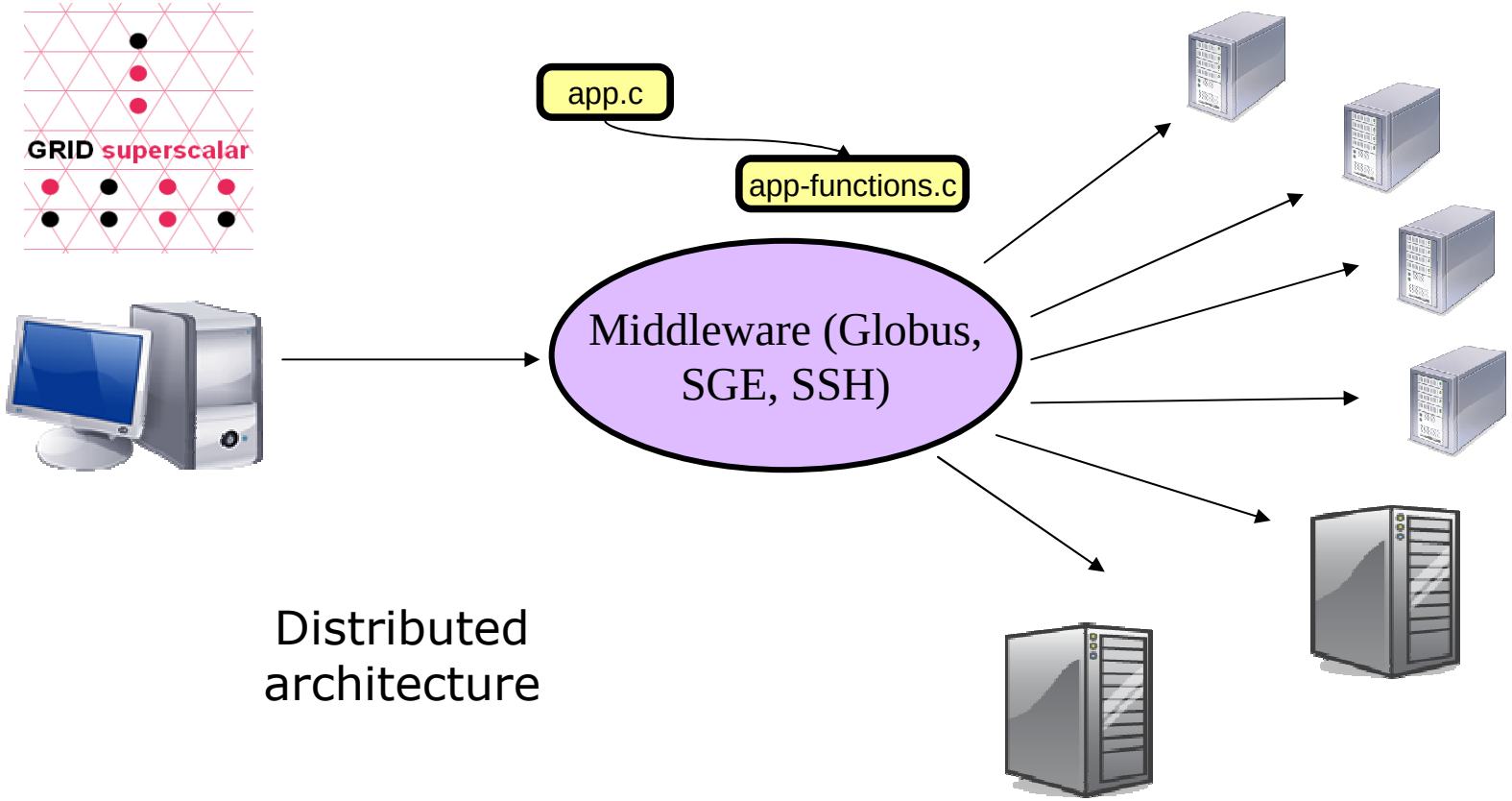
```
void matmul(char *f1, char *f2, char *f3)
{
  getBlocks(f1, f2, f3, A, B, C);
  for (i = 0; i < A->rows; i++) {
    for (j = 0; j < B->cols; j++) {
      for (k = 0; k < A->cols; k++) {
        C->data[i][j] += A->data[i][k] * B->data[k][j];
  putBlocks(f1, f2, f3, A, B, C);
}
```

app.c

app-functions.c

Local scenario

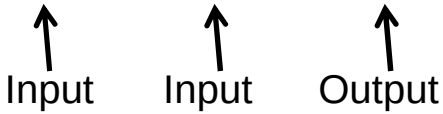
Design


GRID superscalar
app.c
app-functions.c
Middleware (Globus, SGE, SSH)
Distributed architecture

Block matrix multiplication

*void matmul(char *f1, char *f2, char *f3)*

Sequential code prototype



interface MATMUL { void matmul (in char f1, in char* f2, inout char* f3);};*

IDL File

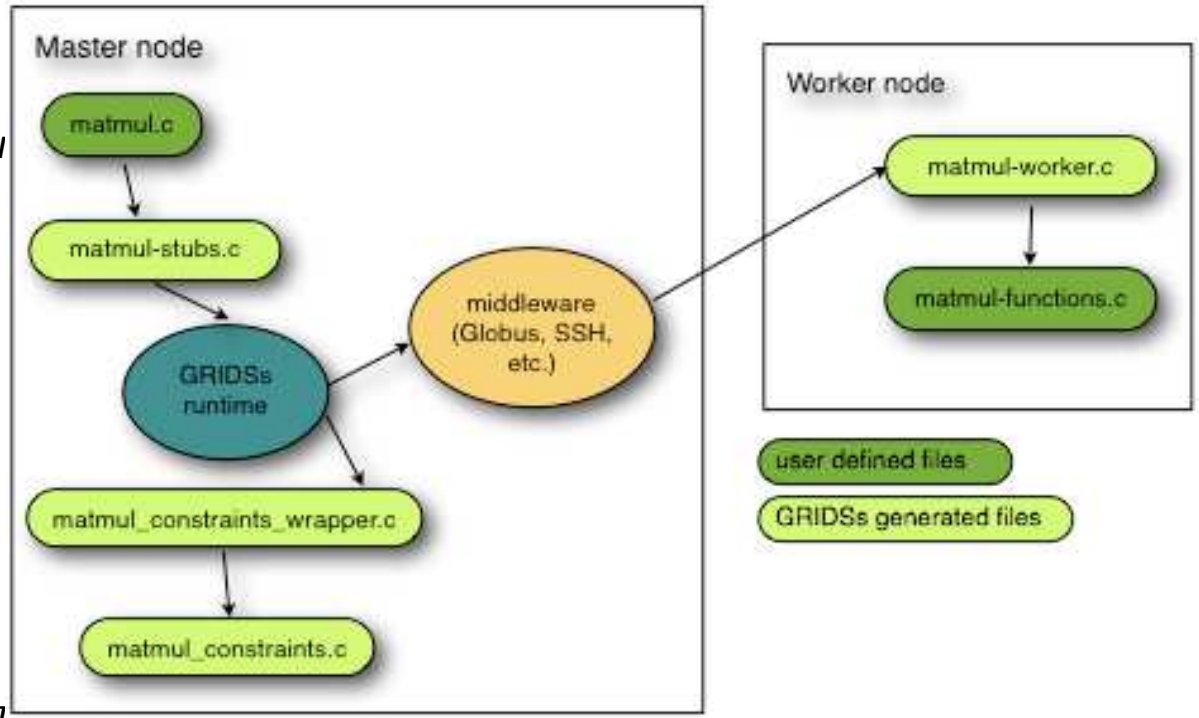
```
GSMaster.On();
for (int i = 0; i < MSIZE; i++) {
        for (int j = 0; j < MSIZE; j++) {
                for (int k = 0; k < MSIZE; k++) {
                        matmul(A[i][k], B[k][j], C[i][j]);
                }
                 }
        }
GSMaster.Off(0);
```

Master code

Block matrix multiplication

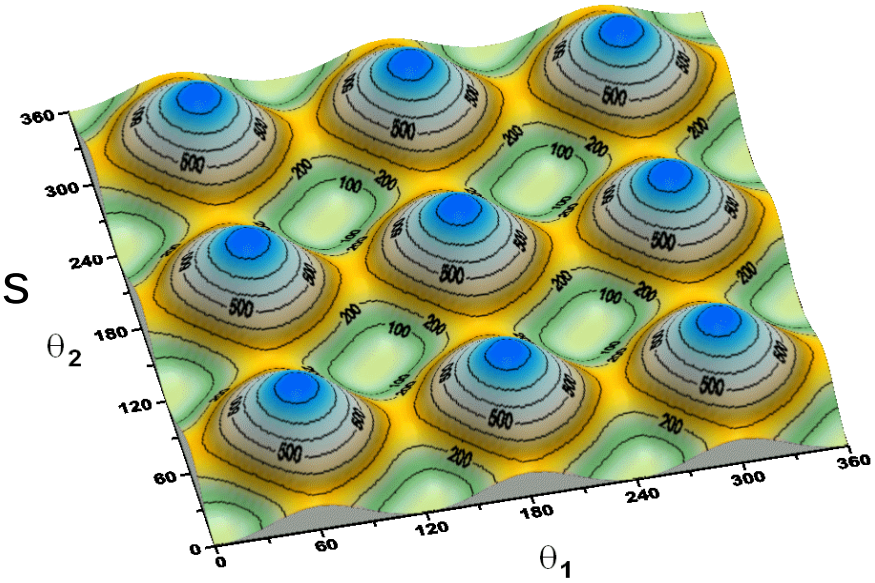
```
void matmul(char *f1, char *f2, char *f3){
        block *A;
        block *B;
        block *C;
        A = get_block(f1, BSIZE, BSIZE);
        B = get_block(f2, BSIZE, BSIZE);
        C = get_block(f3, BSIZE, BSIZE);
        block_mul(A, B, C);
        put_block(C, f3); //A and B are sou
        delete_block(A);
        delete_block(B);
        delete_block(C);
}

static block *block_mul(block *A, block *B,
/* Pre: The three parameters must exist */
int i, j, k;
for (i = 0; i < A->rows; i++)
 for (j = 0; j < B->cols; j++)
  for (k = 0; k < A->cols; k++)
    C->data[i][j] += A->data[i][k] * B->data[k][j];
 return C;
}
```

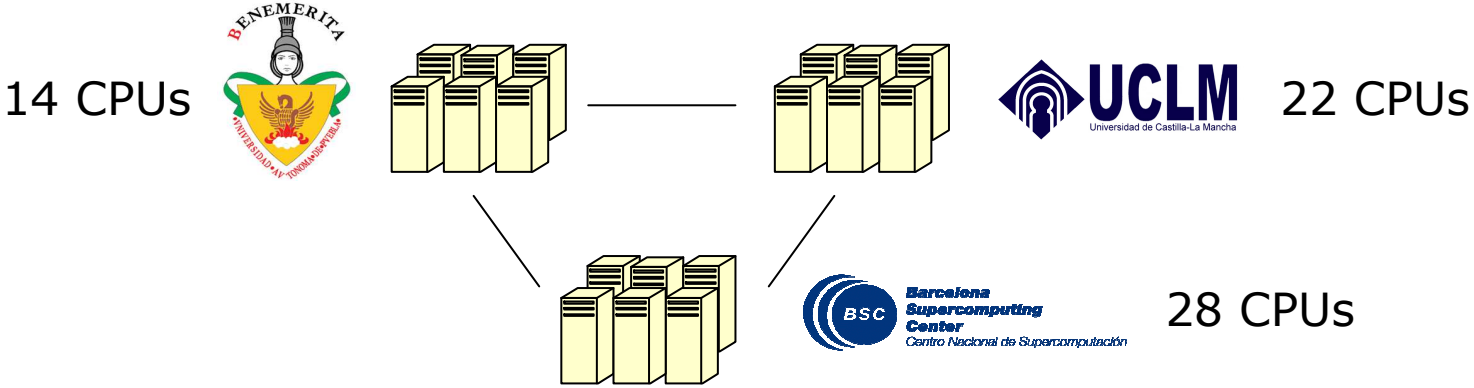
Worker Code



Successful stories Mapping of molecular potential energy hypersurfaces

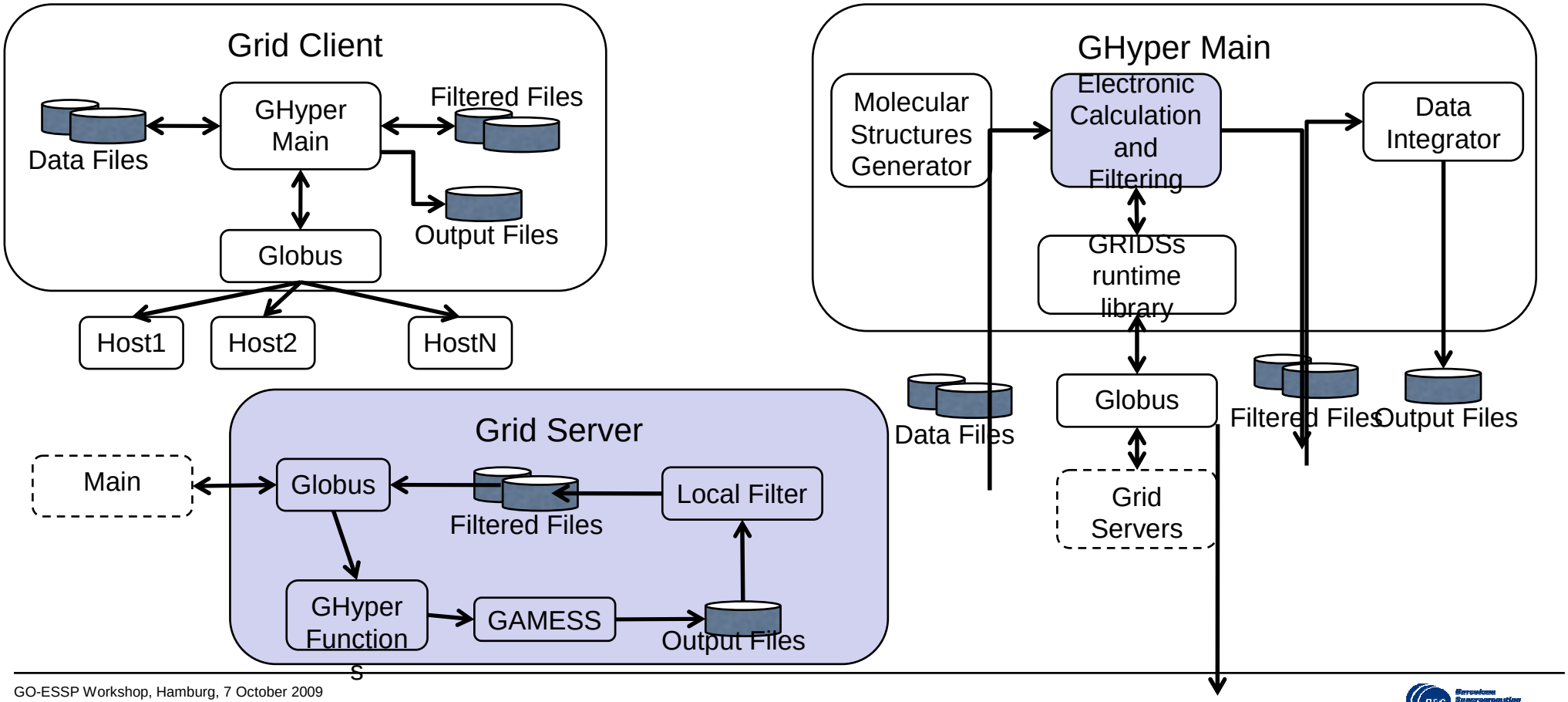
- Total execution time: 17 hours
- Number of executed tasks: 1120
- Each task between 45 and 65 minutes



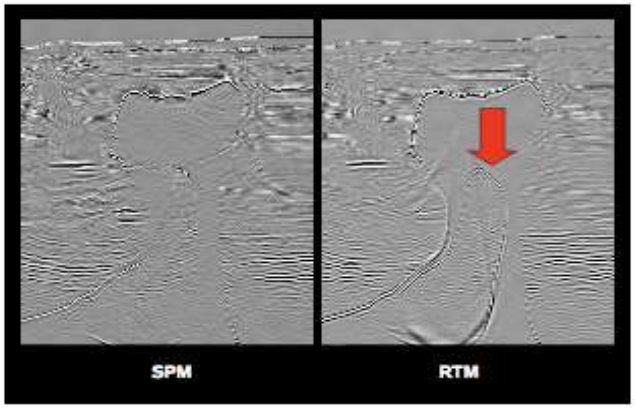


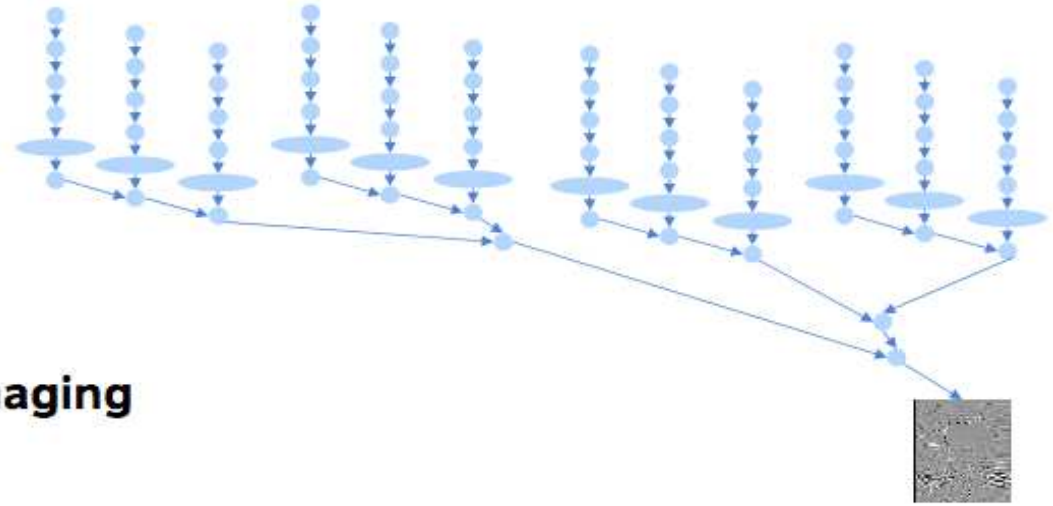
Successful stories: Mapping of molecular potential energy hypersurfaces

1. Molecular structure generation: this step involves the generation of the set of molecular structures defining the potential energy hypersurface. **The result is a large number of input data files**.
2. Electronic structure calculation: one evaluation with the electronic structure package is executed for each of the data files generated in step 1. Since the output of each of these evaluations is a large output file, a **filtering process** that obtains the required information (molecular coordinates and total energy) is applied.
3. Data integration: the data generated by each of the calculations in step 2 is integrated in a single ASCII file.



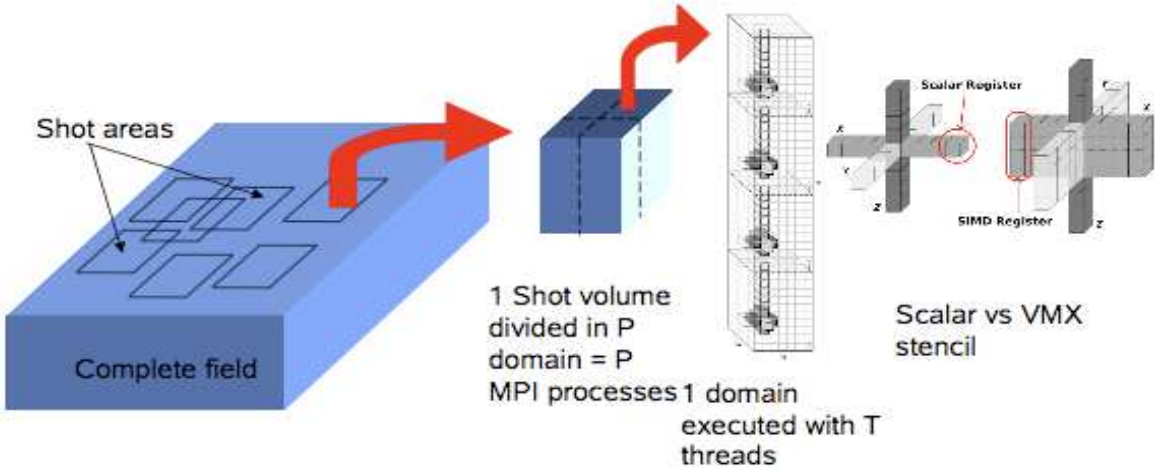
GRIDSs + MPI + OpenMP: Reverse Time Migration (RTM)



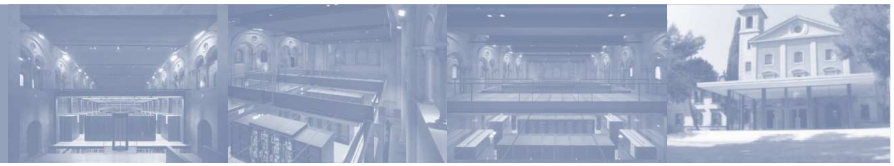


Only RTM produces proper subsalt imaging
Computationally more intensive

- 1 GRID superscalar application per image
  - 350,000 – 500,000 tasks per image
- Domain Decomposition (MPI) to process one shot between several blades
- Threads
  - OpenMP to execute one MPI process per JS21blade
- SIMD capabilities
  - VMX code



The runtime: Replica management

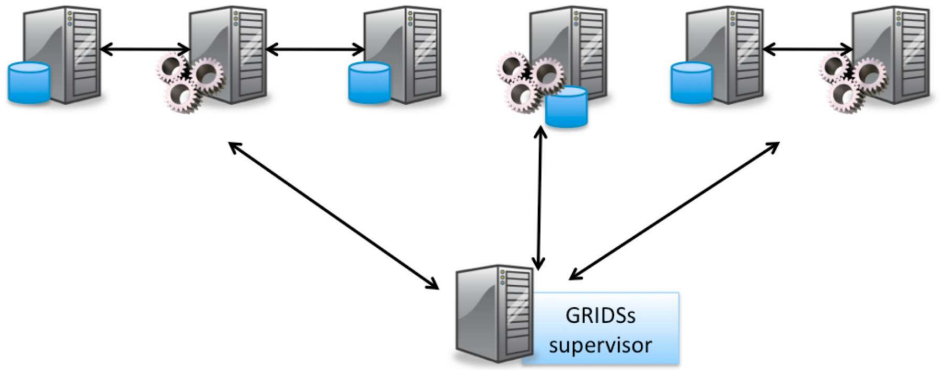


- GridSuperscalar can have its own replica management
  - Applications see the directory tree of their matter, but files may be in any node
    - Hopefully replicated
- Potential benefits:
  - Keep results obtained by one computation to be used by a different one
    - Results are kept in the node that computed them, but are accessible from any node
  - Results reused by many jobs can be stored in all nodes that used them
    - Increases the probability of running a job without having to move the data
  - All is transparent to the applications (as long as the same names are used)

IS-ENES Activities

JRA4, task 3: operational services

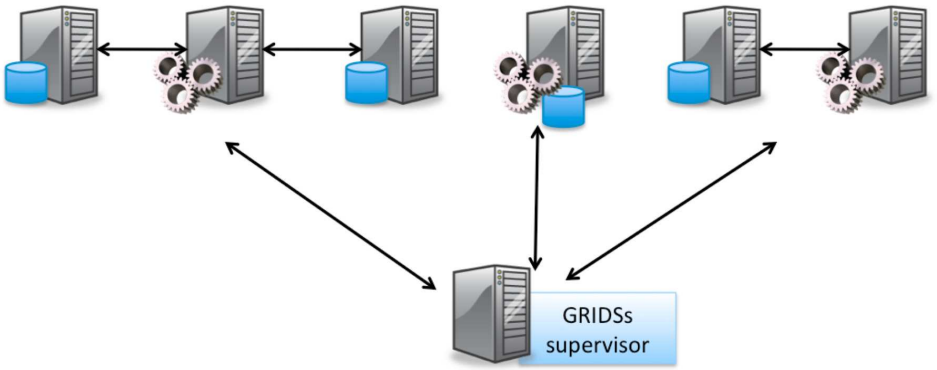
- Definition of an initial testbed
- Multimodel ensemble mean of CMIP5 data held at BADC and WDCC, using CDO regridding; evaluation of netcdf4 compression
- Integration in GRIDSs of the dynamic status of the network and monitoring services (vERC service monitoring, NA2)
- Fortran bindings, replica management and intermediate files handling
- Management of the results between different executions



IS-ENES Activities

JRA4, task 3: operational services. OPEN ISSUES

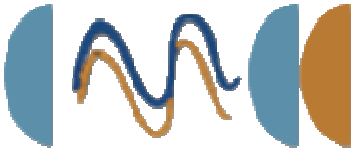
- Relation to ESG
- Data acces (wget, OPeNDAP, GridFTP..)



IS-ENES Activities

NA2 Task4: "Prototype ESM Grid environment"

- Use of the same testbed of JRA4
- Pipelining of the execution of the Echam5 standalone model and of the post-processing of the results (CMCC)



Euro-Mediterranean Center for Climate Change

IS-ENES Activities

Euro-Mediterranean Center for Climate Change

Atmosphere ECHAM5

Model output

GRIB binary format (WMO)

```
Indicator Section
Product Definition Section
Grid Description Section
Bit Map Section
Binary Data Section
End Section
```

Post-processing

user namelist

AFTERBURNER

netCDF (network Common Data Format)

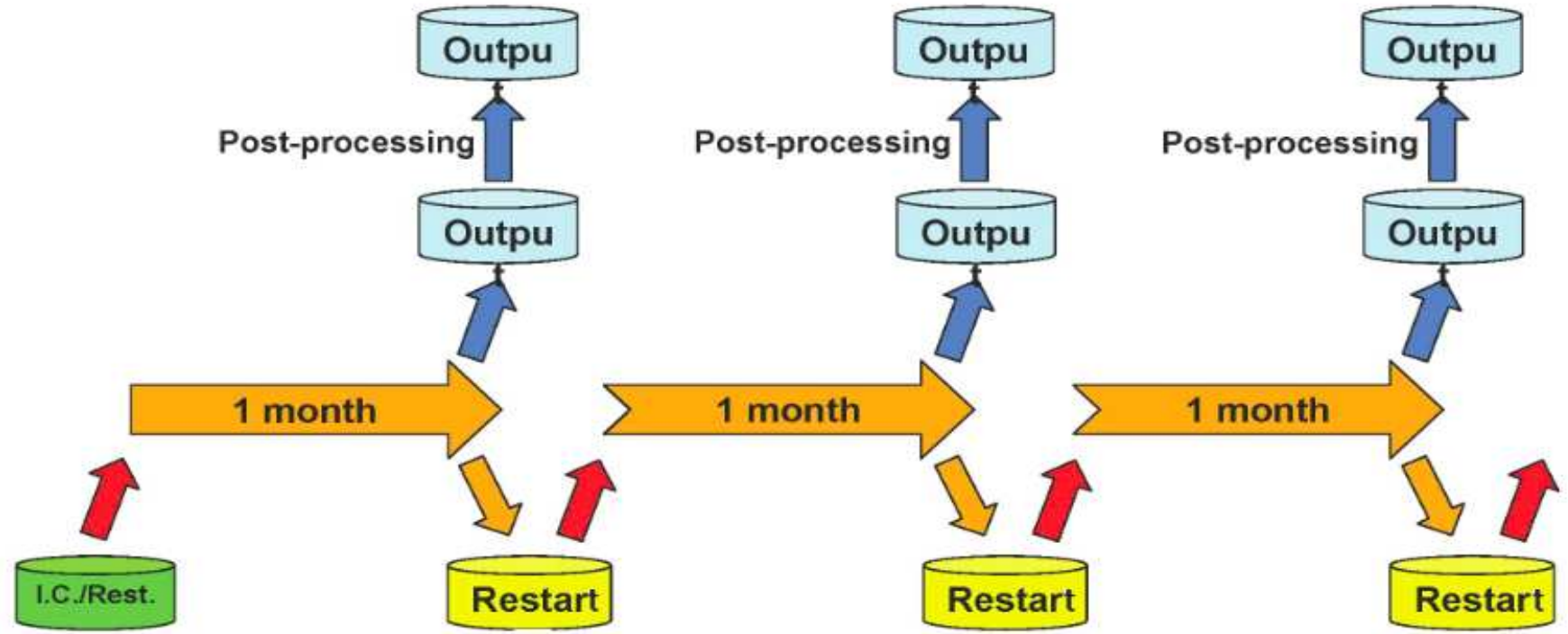
binary format:
- Self-Describing.
- Portable.
- Direct-Access.

CMOR Climate Model Output Rewriter

PCMDI (Program for Climate Model Diagnosis and Intercomparison)

netCDF-CF Climate and Forecast Metadata Convention

IS-ENES Activities



Q&A

http://www.bsc.es/grid/grid_superscalar

daniele.lezzi@bsc.es